Robust and Accurate Vectorization of Line Drawings

Xavier Hilaire, Karl Tombre

Robust and Accurate Vectorization of Line Drawings

Xavier Hilaire and Karl Tombre

Abstract—This paper presents a method for vectorizing the graphical parts of paper-based line drawings. The method consists of separating the input binary image into layers of homogeneous thickness, skeletonizing each layer, segmenting the skeleton by a method based on random sampling, and simplifying the result. The segmentation method is robust with a best bound of 50 percent noise reached for indefinitely long primitives. Accurate estimation of the recognized vector's parameters is enabled by explicitly computing their feasibility domains. Theoretical performance analysis and expression of the complexity of the segmentation method are derived. Experimental results and comparisons with other vectorization systems are also provided.

Index Terms—Document analysis, graphics recognition and interpretation, vectorization, curve segmentation, performance evaluation, line drawings.

1 Introduction

The automated conversion of paper-based documents into a set of features which can be edited and stored by some document management software is a problem that has a long history and which has received considerable attention during the last decades. In the case of line drawings, the aim is usually to convert the graphics represented by pixels into vectors and other simple geometrical features. This problem of *vectorization*, i.e., raster-to-vector conversion, has been at the center of research in graphics recognition for many years already. A number of techniques have been proposed to solve this problem, leading to commercial software packages as well as to research prototypes. All these systems provide quite acceptable results on simple, moderately noisy documents, but recognition rates for poor quality and huge documents such as technical drawings hardly exceed 80 percent in practice, leading to high postvectorization editing costs [7], [33].

Several reasons may explain this. First, it is very unlikely that a "blind" segmentation process can provide satisfying results, not only because the optimal solution for the segmentation is usually not known, but also because it is generally not unique. Introducing ad hoc rules or contextual expert knowledge to obtain better quality solutions therefore appears mandatory, and several authors have explored this pathway [5], [22], [27], [40]. However, due to a potential combinatorial explosion, such approaches may only provide limited improvements and still rely on a high quality low-level vectorization. Also, both the robustness and scale invariance of most of the cited methods are not known and only little (if any) theoretical analysis of their performances have been established so far. At last, a crucial point in most existing methods is the requirement to tune an important set of parameters, often set on completely empirical bases.

• *The authors are with LORIA, 615 rue du Jardin Botanique, 54602 Villers-lès-Nancy, France. E-mail: xhilaire@free.fr, tombre@loria.fr.*

Manuscript received 25 Jan. 2005; revised 20 Aug. 2005; accepted 4 Oct. 2005; published online 13 Apr. 2006.
Recommended for acceptance by V. Govindaraju.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-0053-0105.

The context in which the present work was initiated added another constraint on us, that of high accuracy. The raster-to-vector conversion method we designed had to be used for recovering the dimensions and positions of the different components of architectural drawings with high precision. It was therefore not enough to detect vectors; their position had to be very accurate with respect to the original drawing too.

All this convinced us that a *robust* and *accurate* vectorization method, involving only a limited set of parameters [43] and including a theoretical analysis of its behavior, would bring a significant contribution to the field. This paper aims at presenting such a method. Like most raster-to-vector conversion systems, it is currently limited to the detection of straight line segments and circular arcs, the two mostly used shapes in technical documents. There is, however, no absolute limitation in the proposed approach which would prevent it from detecting other kinds of primitives, although we suspect it to become prohibitively costly in terms of computational complexity.

The method has been implemented and tested on a wide variety of line drawings. In particular, we have made use of the database set up for the international graphics recognition contests organized during the IAPR international workshops on graphics recognition [7], [28], [50] to provide comparative performance evaluations with respect to four other systems. A probabilistic analysis of the method's behavior is also provided.

The remainder of this paper is organized as follows: In Section 2, we briefly review some other proposed approaches and point out their main weaknesses. In Section 3, we provide the necessary material for skeleton-based vectorization. In Section 4, we describe the details of our method and give a theoretical analysis of its behavior. Section 5 details the complexity of the method. Experimental results and performance comparisons are provided by Section 6, and the paper closes with a discussion about lessons learned and further perspectives of this work in Section 7.

2 Related Work

Most vectorization methods consist of several processing steps, including *finding the lines* in the original raster image, *segmenting the lines* found into a set of vectors and arcs, and performing various postprocessing treatments, to find better positions for the junction points, to merge some vectors and remove some others, etc.

As we mentioned in Section 1, postprocessing mostly consists of adding some kind of contextual knowledge. This includes simple heuristics to correct the result, setting junctions straight, merging those which are close to each other, reconnecting lines split up by a missing pixel [5], or taking into account the nature of the drawing to progressively simplify it [40]. Although such approaches can significantly improve raw results, they tend to introduce a number of additional thresholds and parameters. Hence, they will not be considered here because of our need for a robust and noncontextual method with as few parameters and thresholds as possible. In this section, we will therefore concentrate our review of literature on the two first steps in vectorization.

2.1 Finding the Lines: Raster-to-Vector Conversion

Raster-to-vector methods may be roughly classified into three families, depending on the basic technique they rely on: parametric model fitting, tracking and matching opposite contours, and skeletonization.

Parametric model fitting consists in using a line model to detect the lines present in the image. The most general and best-known technique for that is the Hough transform, which has nevertheless not been widely used for raster-to-vector conversion, except some attempts by Dori [10], Kultanen et al. [25], Yamada [53], and Song and Lyu [38]. These methods have only been evaluated on simple or very specific cases. The Hough transform is a global and additive transformation of an image. It does not guarantee that two noisy discrete lines having very close respective slopes won't mix and map into a dummy peak in parameter space. It is time efficient, but memory inefficient. However, Song and Lyu have recently presented a vectorization making use of the Hough transform, but adapted to the analysis of line drawings. Their idea is to perform a subsampling of the image by finding interest points through a horizontal and a vertical scan of the image. Only these points are used in the Hough vote. In addition, the neighborhood of the points is analyzed to only vote for plausible angles. All this reduces the computation time and memory needs of the method and yields promising results. Still, the methods cited have been designed to extract lines only, not lines and circular patterns.

Other parametric models have been applied to domain-specific applications such as finding staff lines [8] or regular lines in forms [54]. This is related to vectorization methods using higher-level contextual knowledge and, therefore, not relevant for the present work, which aims at designing a general method without a priori contextual knowledge about the positions or orientations of the lines and arcs.

Tracking and matching opposite contours is an alternative technique that was tested in the late 1980s [1], [37]. It provides correct and accurate results if the junctions are simple. However, it is noise sensitive and fails when it comes to match complicated structures, even at the price of an expensive computation time [43].

Skeletonization definitely remains the most widely used technique for raster to vector conversion [4], [20], [24], [31]. The basic idea is simply to compute some kind of medial axis of the shape to be vectorized by thinning it or by looking for the local extrema in its distance transform. This relies on the assumption that the shapes to be vectorized are elongated shapes. Once the skeleton has been computed, the problem of vectorization is reduced to that of segmenting a 2D discrete curve into meaningful features.

Unfortunately, the problem is complicated by the crossings and junctions between thick shapes, as parabolic arcs appear in the continuous space for usual crossings involving thick segments [3] regardless of the type of skeletonization methodology used (for an overview of thinning methodologies, see [26], and for nonthinning-based methods, see [48]). Most of the existing methods have only limited ability to detect such parabolic arcs; they approximate them by vectors, which obviously weakens the quality of the solution and is confusing for higher level treatments (namely, recognition and analysis tasks). Early attempts to overcome this drawback were performed by Hori and Tanigawa [19], who proposed to combine skeletonization with contour matching. The critical point in their method resides in the limitations of matching at the junctions. Another approach is that of Janssen and Vossepoel [20], who propose an iterative algorithm based on maximum threshold morphology to adjust the vectors' extremities. Their method performs well on simple junctions, but has the main drawback of relying on local adjustments, which makes it sensitive to noise. Sensitivity to noise is also a critical point in SPV, proposed by Dori and Liu [11]. Their method acts as a simplified and fastened skeletonization operator, as the orthogonal zig-zag walk only visits a few pixels on the shapes' boundaries. Junctions are detected by the presence of an abnormal border-to-border length with respect to the average length observed during the walk; as a result, a junction involving only small segments might be skipped. The method also requires the settings of more than 10 thresholds.

All the above methods have only limited chances to properly process junctions and are noise sensitive. Another more complicated problem is that of extracting circular arcs from the image [21], [49]. This is generally achieved by testing whether a subset of adjacent segments can constitute a circular arc or not; scalability and noise sensitivity of such methods have unfortunately not been examined.

2.2 Curve Segmentation

With respect to the segmentation of the 2D curves resulting from the previous step, the literature offers a considerable number of methods which can be classified in two categories. First, there are methods using curvature estimation to detect critical points on the curve [41], sometimes taking scale space into account [2], [30]. Such methods perform well on curves that are not too noisy and whose scale variation is reasonable, but do not explicitly guarantee any upper bound on the most usual error metrics (such as the integral square error, Haussdorf distance, etc.). Very soft transitions between a pair of lines or lines and circles with very large radii, which are commonly found in technical drawings, make such methods unsuitable for this class of documents.

Second, some methods operate directly on the curve, either by minimizing a particular error measure or by splitting or breaking the curve as long as the local or global error remains too high. Such methods are known as polygonal approximation methods [35] and are more suitable to the problem of

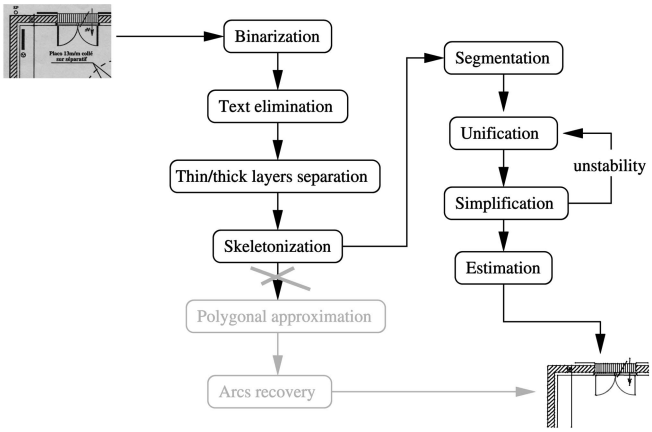


Fig. 1. Overview of our vectorization method with changes with regard to [13] emphasized.

vectorization because of the guarantees they provide. The most usual criterion used by these methods is that of the maximum distance between the curve and the segment [36]. This leads to recursive splitting of the curve at the maximum deviation points until all segments are valid approximations. As the position of the extrema points of the segments tends to be constrained by the initial pixel positions, it makes sense for the method to be followed by a fitting phase, where each segment is displaced to best fit the original curve. A second possible criterion is that of the algebraic area of the surface between the curve and the segment [47]. As this area can be computed iteratively, very time-efficient iterative methods have been proposed. A problem with the approach is that it tends to displace the angular points, as the method only detects a change of general direction after having followed several points past the true angle.

However, whatever criterion is chosen, these approximation methods are still lacking with respect to the requirements of the vectorization problem, as a good vectorization method should simultaneously extract circular arcs and segments, scale invariant and robust to digitization noise as well as to text potentially attached to the graphics, and should not represent any of the parabolic arcs that appear in a crossing of primitives as a result of the skeletonization process.

3 Background

The vectorization method we propose improves on that of Dosch et al. [13] and is fully summarized in Fig. 1. The reader must note that the first four stages of this method (binarization, text elimination, thin/thick layers separation, and skeletonization) should therefore not be considered as a contribution of the present work. They are recalled here for the sole sake of clarity.

3.1 Binarization and Filtering

We regard binarization as an optional stage, as most of the source images to be analyzed are directly available in binary format. In our system, the binarization stage is followed by an elementary filtering procedure that fills the holes that the graphics might carry, removes blobs that may have appeared outside of them, and performs a morphological closure. It is possible to obtain a bound on the surface of spurious holes by resorting to Kanungo et al.'s document degradation model [23], and to the following result, proven in [17] (keeping Kanungo et al.'s notations):

Lemma 1. *The expectation of the size of a fake hole (due to degradation) is lower than*

$$\frac{\alpha_0 e^{-\alpha} + \eta}{\left(\alpha_0 e^{-\alpha} + \eta - 1\right)^2}.$$

The proof is trivial and is not reproduced here. A similar result may be obtained for blobs.

3.2 Text Elimination

The kind of documents we have to process often contain a text layer which is not to be vectorized. In addition, this layer may interfere with the segmentation of the graphics layer into curves when text touches the graphics. We therefore use the text/graphics separation method of Tombre et al. [45], which is based on a method first proposed by Fletcher and Kasturi [16], with a number of improvements. One of these is the ability to retrieve in the text layer characters connected to graphics, provided they belong to a string for which at least some characters are disconnected and have thus been detected as a "string seed." As we will see in Section 6.3, under certain conditions, our method is actually capable of retrieving the correct segments and arcs even when the assumption of string seed presence does not hold.

3.3 Thin/Thick Layer Segmentation

Line drawings can be made by the superimposition of several layers with different thicknesses, each layer carrying some specific information. In that case, our system offers the option of working independently on each thickness layer. In our implementation, we use mathematical morphology operations to perform thin-thick segmentation: An erosion is followed by a partial geodesic reconstruction [13]. Alternatively, this could also be done through segmentation of the distance transform.

Our method actually does not need this and is able to correctly segment a line drawing with different line thicknesses.

3.4 Skeletonization

There are many skeletonization algorithms [26] either based on the iterative deletion of the most outer pixels or on a distance transform and medial axis computation. In our vectorization system, we use di Baja's skeletonization algorithm [9] for three simple reasons. First, the metric used in [9] is the (3, 4)-chamfer distance, which effectively induces a metric over $\mathbb{Z}^2$ [42]. This means that given a discrete thick curve in $\mathbb{Z}^2$, we can expect its skeleton to coincide with a thin curve of the same nature within a gap of one pixel. Second, the skeleton is reversible and the computation of the whole distance transform will allow us to estimate the thickness of each extracted shape, as explained further in Section 4. Third, the skeleton is robust. The detection of saddle points, the computation of the medial axis from the distance map, and its reduction to unit width using positive gradients as described in [9] ensures that skeletal points will always be chosen in the same manner whenever a choice is possible. In particular, adding noise points to the boundaries of a shape does not generally lead to changes in the resulting skeleton [17].

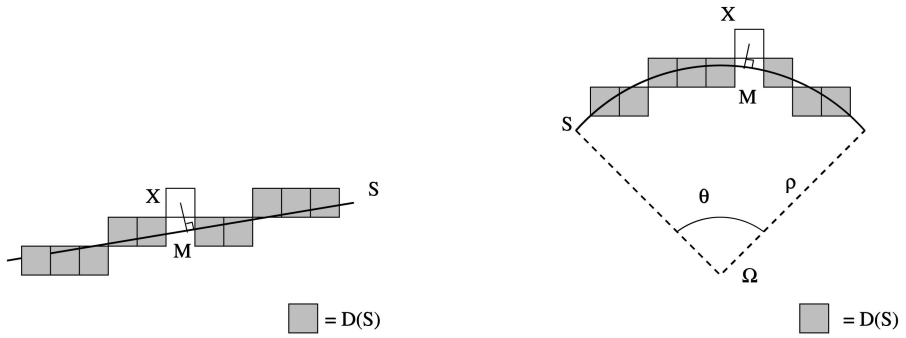


Fig. 2. Definition of fuzzy primitives: (a) Segment. (b) Circular arc. White pixels, which do not belong to $D(S)$, are discarded and replaced by the proper pixels once the primitive is identified.

4 Proposed Vectorization Method

4.1 Segmenting the Skeleton

4.1.1 Hypothesis, Definitions, and Notations

We assume the documents to be preprocessed as described in Section 3 until skeletonization. If the option of segmentation into different thickness layers has been chosen, the following discussion applies to one such layer at a time. The skeleton computed with di Baja's algorithm is connected. For that reason, we will only have to deal with connected discrete arcs and skeletal branches, whose definitions are recalled below.

Definition 1. *A discrete simple arc (DSA) is a sequence* $((x_1, y_1), \ldots, (x_n, y_n))$ *of points of* $\mathbb{Z}^2$ *such as* $1 \leq |x_{i+1} - x_i| + |y_{i+1} - y_i| \leq 2$ *for* $i = 1, \ldots, n-1$. *Furthermore, if* $|x_n - x_1| + |y_n - y_1| > 2$, *the DSA is said to be open.*

Definition 2. *A discrete skeletal branch (or skeletal branch, for short) is a DSA whose points* (x_i, y_i) *are skeletal points and such that* (x_1, y_1) *and* (x_n, y_n) *are skeletal endpoints. A skeletal endpoint is a skeletal point having either exactly one or at least three 8-connected neighbors.*

In the sequel, we will deal with segmentation of skeletal branches only. It is assumed that an upper bound f of the line thickness is known and that the skeleton has been pruned using this value.

Let $\mathcal{C}$ be a skeletal branch; we denote by $\mathcal{C}[k]$ its kth pixel and by $\mathcal{C}[i..j]$ the subarc of $\mathcal{C}$ containing pixels $\mathcal{C}[i]$ to $\mathcal{C}[j]$, $i \leq j$. The Euclidean distance is denoted by $d(.,.)$ and $|.|$ denotes either the length (in number of pixels) or cardinality. For any *continuous* shape S of the plane, we denote by $D(S)$ its digitization according to the OBQ[1] scheme (the shape is assumed to be a part of the frontier of a closed, planar object). As stated in Section 1, we are motivated by a robust segmentation of the skeleton. We therefore propose to segment any skeletal branch $\mathcal{C}$ using *fuzzy segments* and *fuzzy circular arcs*, whose definitions follow:

Definition 3. *A DSA* $\mathcal{C}$ *is a fuzzy segment (FS) if there exists a straight line* S *of the real plane such that*

1. $|D(S) \cap \mathcal{C}| \geq |\mathcal{C}|/2$.
2. $\forall X \in \mathcal{C}, d(X, S) \leq m$, *where* m *is a parameter to specify.*

1. Object Boundary Quantization.

Definition 4. *A DSA* $\mathcal{C}$ *is a fuzzy circular arc (FCA) if there exists a circular arc* S *of the real plane with radius* ρ *and opening angle* θ *such that*

1. $|D(S) \cap \mathcal{C}| \geq |\mathcal{C}|/2$.
2. $\forall X \in \mathcal{C}, d(X, S) \leq m$, *where* m *is a parameter to specify.*
3. $\theta \geq \theta_{\min}$ *and* $\rho \in [\rho_{min}, \rho_{max}]$, *where* $\theta_{\min}$, $\rho_{\min}$, *and* $\rho_{\max}$ *are constants to specify.*

The underlying idea in both definitions is that we accept a given chain of pixels as a segment or as an arc, provided we can find a continuous shape S around which pixels stand closer than a certain distance m (see Fig. 2). For circular arcs, we furthermore require that S satisfies the validity criteria imposed by $\theta_{\min}$, $\rho_{\min}$, and $\rho_{\max}$. At last, we require that at least half of the pixels of a candidate arc coincide with the digitization of S, which is the largest amount of invalid data we can allow without possible confusion (in the Cramer-Rao sense).

4.1.2 Main Procedure

The pseudocode for the main procedure of our algorithm is available in Algorithm 1, in which f is the thickness of the processed layer and L is initially an empty list. The algorithm is based on random sampling [15] and operates as follows (see Fig. 3):

1. Two indexes i and j are randomly chosen along $\mathcal{C}$.
2. The subarc $\mathcal{C}[i..j]$ is then tested (lines 6-9), first as a FS and next as a FCA, in that order. If it is found to be one or another, then the corresponding **extract_** function is called, which finds the lowest and largest integer indexes i_0 and j_0 such as $i_0 \leq i$, $j_0 \geq j$, and $\mathcal{C}[i_0..j_0]$ is still a fuzzy primitive of the right type. The i_b and j_b indexes, which denote indexes of the longest primitive found, are updated accordingly.
3. If there is no primitive instantiable from i, j or if the longest primitive found does not exceed a threshold value T, then we jump to Step 1 to make a new attempt, unless the number of attempts has reached n_{max}.
4. If a primitive could be found by the previous steps (regardless of it being longer than T or not), then it is added to L and the segmentation is reconducted on the remaining parts of $\mathcal{C}$. Otherwise, we terminate in

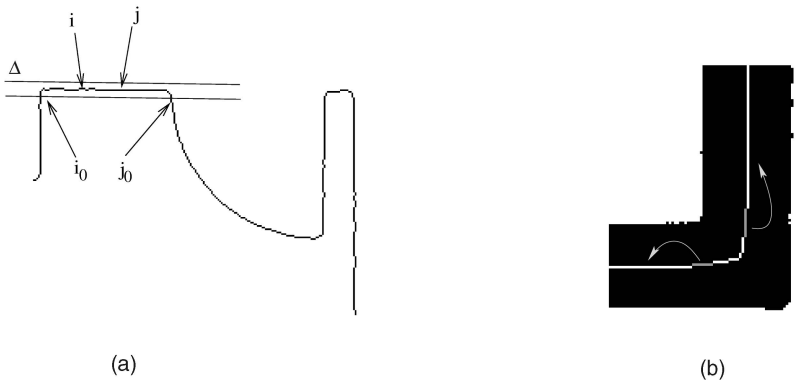


Fig. 3. (a) Illustration of the random sampling procedure used in Algorithm 1. (b) The gray pixels may either belong to the lines or to their closest parabolic arc in the case of an L junction.

failure and apply an alternative method, namely, the split and merge segmentation algorithm [34].

In this procedure, T is a confidence threshold on length upon which we accept any found primitive and break the repeat loop in order to accelerate the process. In other words, if we can find any primitive longer than T, then we retain it; otherwise, we loop at most n_{max} times to retain the longest primitive we found. Setting proper values for T and n_{max} is therefore highly desirable; we detail how to do this in Section 5.3.

Algorithm 1 Pseudocode for the main procedure of our segmentation method.

Function partition ($\mathcal{C}$, m)
Let $l \leftarrow |\mathcal{C}|$, $n \leftarrow 0$, $(i_b, j_b) \leftarrow (0,0)$
repeat
  Randomly choose i and j such as $1 \leq i < j \leq l$
  stop $\leftarrow$ false, $(i_0, j_0) \leftarrow (0,0)$
  if test_segment$(\mathcal{C}, i, j, m)$ **then**
    $(i_0, j_0) \leftarrow$ **extract_segment**$(\mathcal{C}, i, j, m)$
  else if test_arc$(\mathcal{C}, i, j, m)$ **then**
    $(i_0, j_0) \leftarrow$ **extract_arc**$(\mathcal{C}, i, j, m)$
  else
    $n \leftarrow n + 1$
  end if
  if $j_0 - i_0 > j_b - i_b$ **then**
    $(i_b, j_b) \leftarrow (i_0, j_0)$
  end if
until $n \geq n_{max}$ **or** $j_b - i_b > T$
if $j_b - i_b \neq 0$ **then**
  Add the found primitive $\mathcal{C}[i_b..j_b]$ to L
  if $i_b \geq 2$ **then** Segment$(\mathcal{C}[1..i_b], m)$ **end if**
  if $j_b \geq l - 1$ **then** Segment$(\mathcal{C}[j_b..l], m)$ **end if**
else
  Segment $\mathcal{C}$ using the split and merge algorithm and add the result to L.
end if
End

4.1.3 Fuzzy Primitive Test and Extension

The **test_** and **extract_** functions used in Algorithm 1 both require to test whether a given discrete arc $\mathcal{C}$ is a FS or a FCA. According to Definitions 3 and 4, this question may only be answered by building and updating the corresponding feasibility domains in the dual space, which requires $O(n^d)$ time with $d = 2$ for FS and $d = 3$ for FCA.

Rather than using such a costly procedure, we slightly weaken the test by using *linear regression*; that is, we obtain S in Definitions 3 and 4 by fitting a line or a circle to $\mathcal{C}$ using the algebraic least squares method. This approach is justified by the two following lemmas, both proven in [17]:

Lemma 2. *Let $\mathcal{C}$ be an FS or FCA, $\hat{T}$ be the estimate shape of $\mathcal{C}$ using the least square method, and $\hat{\mathcal{C}}$ be the discrete arc obtained by rounding each point of $\hat{T}$ to the nearest integer couple in $\mathbb{Z}^2$. Then, it holds that*

$$\frac{|\mathcal{C} \cap \hat{\mathcal{C}}|}{|\mathcal{C}|} = 1 - O(1/|\mathcal{C}|).$$

Lemma 3. *Addition of bounded and centered noise to a FS or FCA $\mathcal{C}$ does neither change the conditions nor the values toward which their respective least squares estimates converge as $|\mathcal{C}| \to \infty$.*

Both lemmas ensure that recovering all the valid pixels of a fuzzy primitive is always possible when its length becomes arbitrarily large. We conducted and compared both approaches and found only little difference for sufficiently large primitives (length $\geq$ 10 pixels). Therefore, the results reported in Section 6 follow the linear regression approach.

4.2 Optimization

The result of the segmentation stage is a set of primitives that approximate the skeleton of a particular layer of the document. This set, however, may not yet be considered as the solution to vectorization for two obvious reasons: 1) we segmented the skeleton of the image, not the image itself; as a result, we still have to get rid of the skeletal arcs that appear in crossings of primitives, and 2) we considered only skeletal branches, which implies that any primitive of the ground truth has been detected as fragmented if it is hit by any other at a crossing.

The optimization stage addresses these problems. It consists of a *simplification* and a *unification* procedure that are applied in turn, independently. Both procedures share a connectivity graph G built immediately after the segmentation stage; this graph has the following properties:

- Its nodes are the primitives found upon completion of the segmentation stage.
- Any of its edges denotes a connection between two primitives. The edges are unoriented.

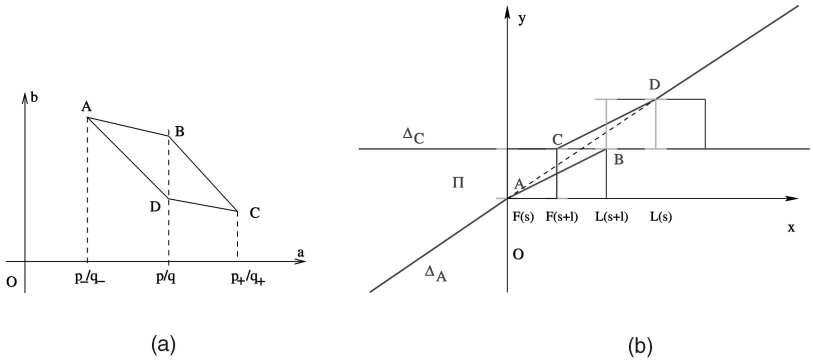


Fig. 4. (a) The domain and (b) the preimage of a discrete segment.

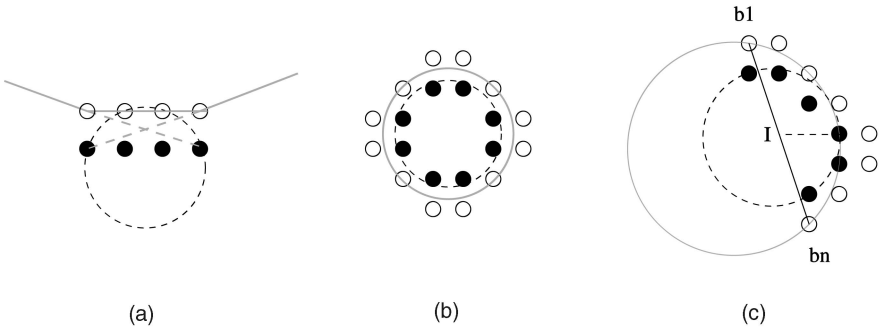


Fig. 5. The three possible forms assumed by the preimage of a discrete circular arc: (a) degenerated case (the circular arc is also a discrete segment), (b) full circle, and (c) regular case.

4.2.1 Simplification of the Skeleton

The *simplification procedure* eliminates the spurious arcs that appeared inside the crossings of primitives during skeletonization. It uses the basic underlying assumption that any discrete arc with length lower than an acceptance threshold f may be removed, provided that the remaining primitives are sufficient to explain and reconstruct the image. To detail this procedure, we first give a few definitions and results from [17].

Definition 5. *The preimage of a DSA $\mathcal{X}$ is the set of all the primitives of the plane whose digitization fully includes $\mathcal{X}$. It is denoted $Pre(\mathcal{X})$.*

Dorst and Smeulders [12] gave a characterization of the feasibility domain of discrete segments in the dual space: It consists of a quadrilateral with at most four vertices (Fig. 4). From their result, it immediately follows that the preimage of a discrete segment can be represented using at most two segments and four half-lines. For discrete circular arcs, we proved in [17] that the preimage could only have three possible shapes, using a lemma from O'Rourke et al. [32]. These shapes are presented in Fig. 5.

We now consider the intersection of two discrete primitives that we define in a particular manner:

Definition 6. *Let $\mathcal{P}$ and $\mathcal{Q}$ denote two discrete primitives, L and R the integer endpoints of $\mathcal{P}$, and $\bar{I}$ the complement of the source image. We say that $\mathcal{P}$* joins *$\mathcal{Q}$* by *L (respectively,* by *R) and denote $\mathcal{P} \vdash_L \mathcal{Q}$ (respectively, $\mathcal{P} \vdash_R \mathcal{Q}$) if and only if there exist two discrete primitives $\mathcal{X}$ and $\mathcal{Y}$ such that:*

1. $\mathcal{P} \subseteq \mathcal{X}$, $\mathcal{Q} \subseteq \mathcal{Y}$, *and* $\mathcal{X} \cap \mathcal{Y} \neq \emptyset$,
2. $(\mathcal{X} \cup \mathcal{Y}) \cap \bar{I} = \emptyset$, *and*
3. $\exists K \in \mathcal{X} \cup \mathcal{Y} : [RK] \cap [RL] = [RL]$ *(respectively,* $[LK] \cap [LR] = [LR]$*),*

where $[xy]$ is the discrete primitive included in $\mathcal{X}$ with endpoints x and y. Note that $\mathcal{X} \cap \mathcal{Y}$ is used because this intersection is not necessarily reduced to a single point in the case of circles. Also, note that K always belongs to a connected component of $Pre(\mathcal{X}) \cap Pre(\mathcal{Y})$; this connected component is called the L-endpoint domain *(respectively,* R-endpoint domain*) of $\mathcal{P}$.*

An illustration of this definition is given in Fig. 6. In each case, the idea is the same: $\mathcal{P}$ and $\mathcal{Q}$ being given, if we can find a piece of discrete shape that completely overlaps $\mathcal{P}$ and permits to reach $\mathcal{Q}$ while remaining inside the image, then we have $\mathcal{P} \vdash_L \mathcal{Q}$ or $\mathcal{P} \vdash_R \mathcal{Q}$, or possibly both, depending on the nature of $\mathcal{P}$ and $\mathcal{Q}$ and on which points the linking piece of primitive stems from. In all cases, Δ_R and Δ_L define the feasible region of the endpoints R and L. In the following, we will always assume that any discrete primitive has two endpoints L and R: The full discrete circle can be viewed as a circular arc whose endpoints are identical, but it can be processed exactly the same way as segments and open circular arcs.

With the above material, we are now ready to explain the **simplify** procedure detailed in Algorithm 3. This procedure operates in two stages. First, for each *long* primitive ($length \geq f$), we seek for similar neighbor primitives that can be reached through a chain of *short* primitives—those *might* be removed. This is done by the **search** function, whose pseudocode is available in Algorithm 2 and which returns the list of all possible paths stemming from a given node n and leading either to a long primitive or to an end of chain of short primitives. To each node that represents a long primitive, the list of these paths is being attached. For any node n, we process all the paths P we find and look at p, the last element of the path P. Several cases may then occur:

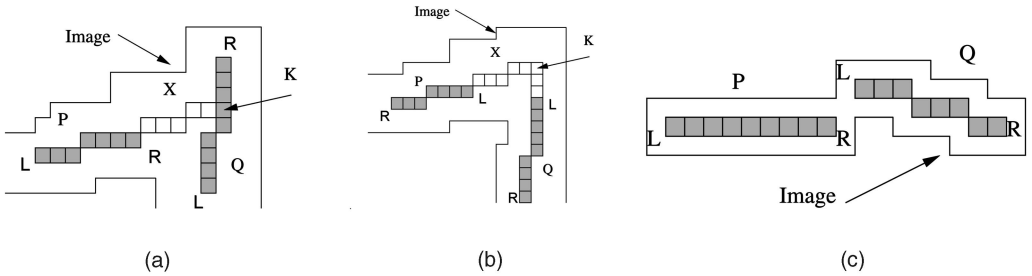


Fig. 6. Illustration of Definition 6. (a) $P \vdash_R Q$, but $P \nvdash_L Q$, $Q \nvdash_R P$, and $Q \nvdash_L P$. (b) $P \vdash_L Q$ and $Q \vdash_L P$, but $P \nvdash_R Q$ and $Q \nvdash_R P$. (c) No joint relation at all.

- p is a short primitive: The edges defined by the path P are simply marked as deletable.
- p is a long primitive and $p \vdash_L n$ (respectively, $p \vdash_R n$): We redefine Δ_L (respectively, Δ_R) as the proper component of $Pre(n) \cap Pre(p)$ following Definition 6. The endpoint L (respectively, R) can then be chosen as any integer point of Δ_L (respectively, Δ_R) in subsequent calls.
- p is a long primitive, but $p \nvdash_L n$ and $p \nvdash_R n$: nothing happens.

As the two first cases lead to a valid intersection, the edges formed by all the found paths between the two primitives are marked for deletion. Finally, all marked edges and all unconnected nodes of G are removed. Note that Algorithm 3 is provided in its simplest form for the sake of clarity and may be optimized in many ways; in particular:

1. The computations of $Pre(n) \cap Pre(g)$ for various n and p, and the information "does $n \vdash_L last(P)$ or $n \vdash_R last(P)$?" can be cached.
2. The redundancies of the deletions can also be avoided by resorting to some proper data structures.
3. Most importantly, if the required endpoints have to be output as *integers*, which is actually the case for the VEC format [6] used in Section 6, then *all the geometric operations* may be handled in $\mathbb{Z}^2$ rather than $\mathbb{R}^2$.

The latter point greatly simplifies the implementation of the method, as handling nonlinear pieces of curves for circles as well as irrational representation and precision issues becomes irrelevant. Our implementation does a little bit better: We allow a rational representation of all points by subsampling the various domain boundaries. All geometric operations are then carried out using standard algorithms on polygons with rational vertices coordinates. It is, however, noticeable that the method we describe is still valid in $\mathbb{R}^2$, but at the price of much costlier algorithms due to precision issues.

Algorithm 2 Pseudocode of the auxiliary **search** function used in the simplification stage.

Function search (n, G)
$L \leftarrow \emptyset$; mark n
for $x \in neighbors(n, G)$ such that x is not marked **do**
  if $length(x) \geq f$ **or** $(T \leftarrow \textbf{search}(x, G)) = \emptyset$ **then**
    $L \leftarrow L \cup \{x\}$
  else
    for $t \in T$ **do**
      $L \leftarrow L \cup \{xt\}$
    end for
  end if
end for
unmark n; return L
End search

Algorithm 3 Pseudocode for the **simplify** procedure of the optimization stage.

Procedure simplify (G)
for $n \in G$ **such that** $length(n) \geq f$ **do**
  for $P \in$ **search** (n, G) **do**
    let p be the last element of path P;
      $remove \leftarrow (length(p) < f)$
    if not $remove$ **then**
      if $p \vdash_L n$ **then**
        redefine $\Delta_L(p)$ according to Definition 6;
        $remove \leftarrow true$
      end if
      if $p \vdash_R n$ **then**
        redefine $\Delta_R(p)$ according to Definition 6;
        $remove \leftarrow true$
      end if
    end if
    if $remove$ **then**
      add edge $(n, last(P))$ to G if it doesn't exist
      mark all edges defined by P for deletion
    end if
  end for
end for
remove all edges from G marked for deletion
remove all unconnected nodes from G
End simplify

To illustrate how Algorithm 3 works, consider the case of the X junction in Fig. 7. The branches in this junction have unequal thicknesses, and we may not expect them to meet at a single point (Fig. 7a). We assume that A, B, C, D are long segments that have been properly recognized during the segmentation stage, whereas a, b, c, d, i, j, k are spurious arcs with length smaller than f (Fig. 7b). We obtain the connectivity graph G of Fig. 7c and, node by node, the following paths: $A \rightarrow \{bB\}$, $B \rightarrow \{cdC, cdkD, cdijD\}$, $C \rightarrow \{dcB, dkD, dijD\}$, $D \rightarrow \{kdcB, kdC, jidC, jidcB\}$. Starting from A, we obtain that A leads to B and that $A \vdash_R B$, so the graph is modified accordingly (Fig. 7d). From B, we obtain two candidates C and D, but only $B \vdash_R C$, so the edges Bc, cd, dC are deleted, the arc BC is added, and G is modified (Fig. 7e). From C, we find three paths, among which one leads to B which has already been processed and the two others lead to D. Since $C \vdash_R D$, we delete Cd, dk, kD, di, ij, jD and insert CD (Fig. 7f). Processing from D is the symmetrical case of processing from C and doesn't change anything. Finally, the remaining, unconnected nodes are simply deleted (Fig. 7g). We therefore obtain the connectivity graph of Fig. 7h and the X junction is

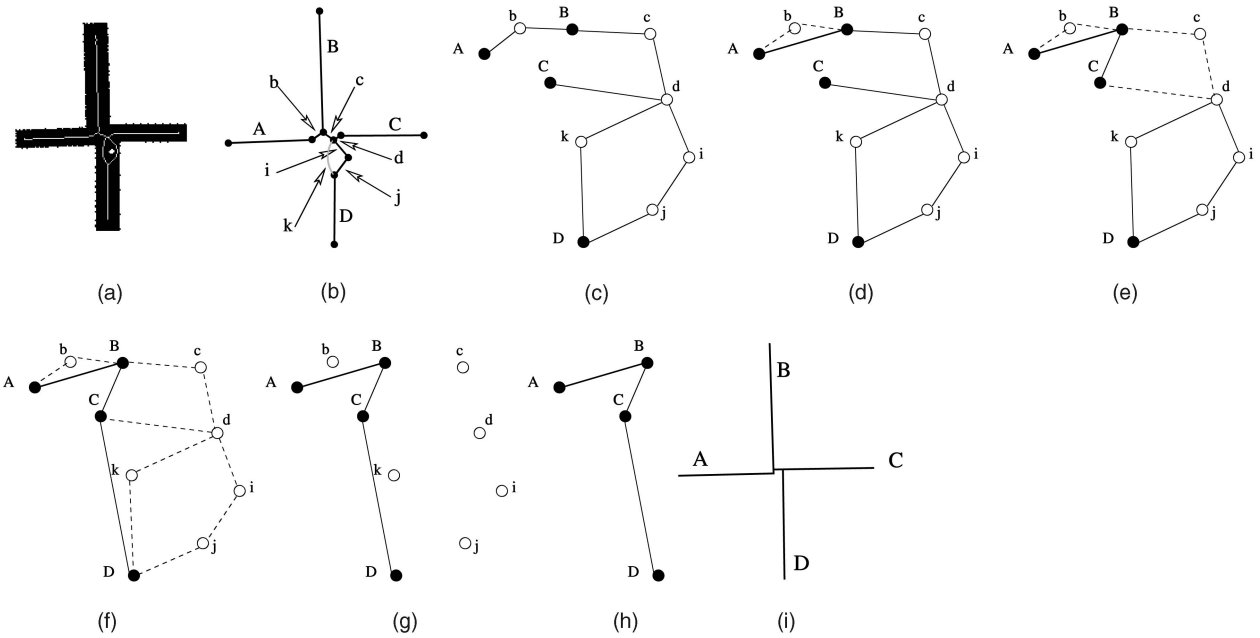


Fig. 7. Processing of an X junction by Algorithm 3. (a) Source image, (b) resulting segmentation, (c) connectivity graph G, (d), (e), (f), (g), and (h) action of Algorithm 3, and (i) the reconstructed junction.

reconstructed as illustrated in Fig. 7i. It is worth noting that reconstructing a given junction generally leads to more than a single solution. It may be interesting to group primitives that overlap each other as much as possible, as we suggested in [18]. It is even sometimes desirable to determine the concurrency of lines through the calculation of the maximal clique of their intersection graph [46], which is an NP-complete problem. The solution we suggest is therefore somewhat arbitrary, but has the advantage to preserve connectivity and is able to find the "carrier" primitives in a junction—primitives can only be grown, never reduced.

4.2.2 Unifying Primitives

Due to the crossing of primitives, we also need to cope with the problem of recovering from primitive fragmentation whenever it occurred. The unification test we use involves Lemmas 2 and 3 and the following domain construction procedure: Let $\mathcal{Q}$ and $\mathcal{P}$ be two adjacent fuzzy primitives; we unify them into a new primitive iff we can find a continuous segment or circle S such as

$$|D(\mathcal{Q}) \cap D(S)| \geq |D(\mathcal{Q})|/2 \text{ and } |D(\mathcal{P}) \cap D(S)| \geq |D(\mathcal{P})|/2. \quad (1)$$

Most of the time, unifying two primitives $\mathcal{Q}$ and $\mathcal{P}$ will result in a primitive S for which $Pre(S) \subset Pre(P) \cup Pre(Q)$. As a result, some concurrency relations established during the simplification stage *may* have to be reconsidered. Indeed, two cases of the figure may occur:

- We find that the endpoint domains of $\mathcal{Q}$ and $\mathcal{P}$ still have a nonempty intersection with $Pre(R)$: In this case, this intersection defines the new endpoint domains.
- We find that the previous intersection is empty: In this case, we simply disconnect the new primitive from those previously attached to its endpoints.

Disconnecting primitives in the latter case results in the processing of two independent primitives and is most of the time desirable (as an example, consider a full circle interrupted by a line). However, it may also become an arbitrary decision in certain cases (Shall an X junction be reconstructed with four segments and a point or only with two segments?). Also, note that there exists an arbitrary order within which primitives eligible for unification can be processed. In our method, we first process those that are the most likely; that is, we maintain a table of all couples of adjacent primitives that satisfy (1) and always process first the pair that maximizes the quantity $|D(S) \cap D(\mathcal{Q})| + |D(S) \cap D(\mathcal{P})|$.

4.3 Final Parameters Estimation

This last stage aims at estimating the parameters of the found shapes and their thickness. It is done once, after all other stages.

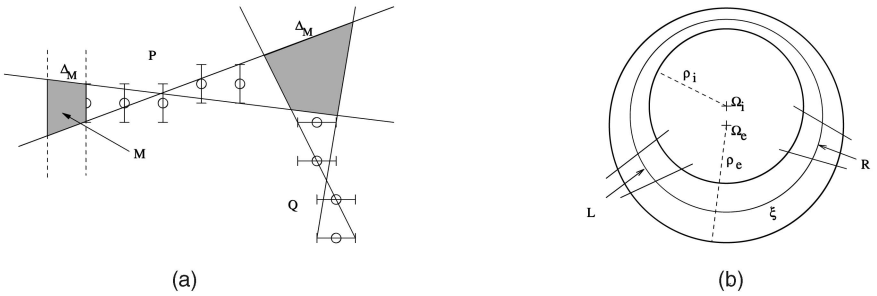
4.3.1 Estimating the Shape Parameters

Estimation of the shape parameters for the case of segments is straightforward from their domains. First, recall from Section 4.2 that, M being L or R, the fact that $\mathcal{P} \vdash_M \mathcal{Q}$ results in a nonempty domain Δ_M (Fig. 8), inside which the endpoint M will have to lie. The optimal coordinates of M are the respective expectations of x and y over Δ_M if we assume a uniform distribution of the feasible points in $\mathbb{R}^2$. In other words,

$$P = \frac{1}{area(\Delta_M)} \left(\int_{D \in \Delta_M} x(D) dD, \int_{D \in \Delta_M} y(D) dD \right).$$

Second, observe from the left of Fig. 8a that a similar domain may also be built in the case of a "free" endpoint M by considering the set of points in the neighborhood of M that don't increase the digitization of the segment (or circular arc).

For circles, Δ_M is bounded by two circles with centers Ω_i and Ω_e (not necessarily identical) and radii ρ_i and ρ_e. The optimal circle is centered in the middle of Ω_i and Ω_e and has radius $\frac{1}{2}(\rho_i + \rho_e)$. For circular arcs, the estimation is slightly more complicated, because the problem is overconstrained.



(a) (b)

Fig. 8. Use of the preimage and endpoint domains to estimate the parameters of (a) segments and (b) circular arcs.

As for circles, we obtain an optimal estimated circle ξ with radius ρ (see Fig. 8b), but the optimal endpoints L and R may not necessarily lie on it. We therefore have two possibilities: Either we set points L and R and seek for a nonoptimal radius or we determine ξ with an optimal radius and seek for nonoptimal L and R points on it. We adopted the latter.

4.3.2 Thickness Estimation

The thickness of the primitives remains quite unreliable information in practice as it is known most of the time with only limited accuracy. Three simple reasons explain this: 1) There is an obvious dependency to the binarization scheme. 2) The shapes are noisy. 3) Even for nonnoisy shapes and a fixed threshold binarization scheme, the theoretical precision is rather poor (for example, any real half-plane may be moved along the Ox or Oy axes of a distance in the interval $]\sqrt{2}/2, 1[$ depending on the plane orientation without affecting its digitization).

The $(3,4)$-distance transform computed during skeletonization provides a straightforward solution to obtain a lower bound of the thickness of a shape. Let I be the digital image and T be the $(3,4)$-distance distance of I. A point $p \in T$ with maximal label v ensures that there exists a discrete ball centered on p with radius v that is fully included in I. Two cases may then occur, depending on whether p is the sole point having the maximal label in its eight-neighborhood to the exception of other skeletal points or not. This test is directly related to the local parity of the thickness. If p is alone (odd local thickness), then v is the value of the thickness with the $(3,4)$-distance. Otherwise (even local thickness), the local description of the shape is a union of balls, in which case, we can deduce the thickness from the diameter of this union and add $3/2$ to the current label's value to correct the estimate. At last, we also have to cope with noise and dummy characters or primitives touching the graphics, which may alter the values of the thickness. We therefore estimate the thickness of each segmented shape with skeleton $\mathcal{C}$ using the following method:

1. For any pixel $p = (i,j) \in \mathcal{C}$, let $E_p = \{(x,y) \in T : \max(|x-i|, |y-j|) \leq 1 \text{ and } T(x,y) = T(i,j)\}$.
2. Collect a set L of values by adding the following value v_p for each $p \in \mathcal{C}$ to it:

$$v_p = \begin{cases} T(p) & \text{if } |E_p| \leq 3 \\ T(p) + 3/2 & \text{otherwise.} \end{cases}$$

3. Compute the median value t of L and retain $2t/3 - 1$ as the thickness.

5 Complexity and Performance Analysis

This section evaluates the complexity of the algorithms involved in our method. We also provide clues to set the T and $n_{\max}$ parameters for the segmentation step (Algorithm 1). We assume an image $h \times w$ pixels wide, containing p connected components. The lowest maximal surface area of these components is denoted by s. The segmentation stage is assumed to process n skeletal branch, each formed of l pixels in the worst case, and N denotes the highest number of primitives carried by at least one of these branches.

5.1 Preprocessing Steps

The first steps, which we "inherited" from previous work (see Section 3), do not present particular high complexity. Noise filtering is essentially linear. Text elimination is based on classical connected component analysis which can also be performed in linear time.

Thin-thick separation is performed by iterating elementary morphological operations a limited number of times. More precisely, let S be the source image from which we wish to extract parts with thickness at least equal to some $e > 0$. If $r = \lfloor (e-1)/2 \rfloor$, the operation consists of iterating r elementary erosions, followed by $r+1$ dilations and intersections with the original image. The computational complexity for separating into two layers is therefore in $O(rhw)$.

5.2 Skeletonization

The time necessary to complete the skeletonization step is bounded by that of the computation of the $(3,4)$-chamfer distance. It turns out to be $O(hw)$ in a sequential scheme [42].

5.3 Skeleton Segmentation

We provide a more detailed analysis of Algorithm 1 in order to establish its complexity and assess its performances. Consider a fuzzy primitive $\mathcal{C}$ with length l. Assuming stochastic noise, let α be the probability that a given point of $\mathcal{C}$ is incorrect. Then, the probability to obtain at least $\lceil l/2 \rceil$ incorrect points in $\mathcal{C}$ is

$$P(n \geq \lceil l/2 \rceil | \mathcal{C}) = \sum_{k=\lceil l/2 \rceil}^{l} \binom{l}{k} \alpha^k (1-\alpha)^{l-k}. \quad (2)$$

An elementary study of the right expression in (2) shows a convergence toward 0 as $l \to \infty$ whenever $\alpha < 1/2$. As a result, there always exists a value of T sufficiently large so that $l > T \Rightarrow P(n \geq l | \mathcal{C}) < \varepsilon$ holds for any $\varepsilon > 0$. Moreover,

TABLE 1
A Few Values of $P(n \geq \lceil l/2 \rceil | \mathcal{C})$ Given l and α

$\alpha \setminus l$	5	10	15	20	25	30
0.45	0.4069	0.4956	0.3465	0.4086	0.3063	0.3552
0.35	0.2352	0.2485	0.1132	0.1218	0.06044	0.06519
0.25	0.1035	0.07813	0.0173	0.01386	0.00337	0.00275
0.15	0.02661	0.009874	0.0006096	0.0002484	1.689e-5	7.078e-6
0.05	0.001158	6.369e-5	1.83e-7	1.134e-8	3.591e-11	2.306e-12

if α is small, then $P(n \geq l|\mathcal{C})$ vanishes rapidly as l increases; more precisely,

$$P(n \geq \lceil l/2 \rceil | \mathcal{C}) = O\Big(\alpha^{\lfloor \frac{l}{2} \rfloor + 1}\Big).$$

Table 1 provides a few values of $P(n \geq \lceil l/2 \rceil | \mathcal{C})$ given l and α. The reader may observe that these values depend on the parity of l, which is no surprise given (2). Of course, the problem of estimating α remains, as we are not supposed to know it before runtime. However, we may see from Table 1 that setting $T \geq 25$ leads to quite acceptable values for any $\alpha \leq 0.4$.

Suppose now that $\mathcal{C}$ is made of N primitives of integer length $l/N < T$. We denote by E_1 the event "i and j belong to the same ground truth Primitive," by E_2 the event "$\mathcal{C}[i..j]$ has more correct than incorrect Points," and by E_3 the event "Algorithm 1 finds a primitive within n_{max} trials."

The **test** and **extract** functions cannot fail if both E_1 and E_2 occur, so we have

$$P(E_3|\mathcal{C}) \geq 1 - (1 - P(E_1|\mathcal{C})P(E_2|\mathcal{C}))^{n_{max}}$$

which immediately leads to

$$n_{max} \leq \frac{\log(1 - P(E_3|\mathcal{C}))}{\log(1 - P(E_1|\mathcal{C})P(E_2|\mathcal{C}))}. \quad (3)$$

Obviously, $P(E_1) = 1/N$ and the expression of $P(E_2|\mathcal{C})$ given by (2) may be assumed to be close to 1. If we neglect it, we can rewrite (3) as

$$n_{max} \approx \frac{\log(1 - \sigma)}{\log(1 - 1/N)} \quad (4)$$

in which $\sigma = P(E_3|\mathcal{C})$ represents the desired probability that Algorithm 1 succeeds after n_{max} trials under the hypothesis made on $\mathcal{C}$. In our system, we use (4) to determine n_{max}. The worst case corresponds to a systematic failure of Algorithm 1 after n_{max} iterations, followed by the application of the split and merge algorithm. Each iteration of Algorithm 1 runs in linear time with the number of pixels carried by the candidate fuzzy primitive and, finally, the complexity of the split and merge algorithm in the worst case is $O(l \log l)$. It therefore follows that the time complexity of the segmentation stage in the worst case is

$$O\left(nl\left[\log l + \frac{\log(1 - \sigma)}{\log(1 - 1/N)}\right]\right).$$

A noticeable fact is that, because of the $\log$ terms, the above expression is exponential with regards to both σ and N. This means that the method we suggest is not suitable for large values of N and that we will always have to tolerate a few chances for failure if we want to avoid irrelevant computational times.

5.4 Optimization

The worst case for the simplification stage occurs when the ground truth is made of n skeletal branches all cutting each other. In that case, the execution time of Algorithm 3 is bounded by $O(n^2)$ (for the path search, the deletion of edges, but also the computation of the domains and their intersections).

In the unification stage, the equation of the potential, unifying primitive is known in constant time, but the fuzzy primitive test requires linear time with regard to the number of pixels to test. The worst case occurs when all the primitives can be unified in a single one, with the additional constraint that only a single segment is integrated at a time (for example, with n adjacent, collinear segments). In that case, the resulting complexity is $O(n^2 l)$.

6 Experimental Results

We evaluated the capabilities of our method over the set of images used during the third, fourth, and fifth IAPR contests on graphics recognition [7], [28], [50], held, respectively, at the GREC 1999, GREC 2001, and GREC 2003 workshops. Whereas a minimal tuning of the different systems' parameters was allowed during the third contest, no preparation and human intervention was permitted at all during the fourth and the fifth. In order to make a fair comparison of our results with those obtained by the contestants, *we strictly conformed to the contests' respective policies during our experiments.* In particular, the default settings we used to obtain the results reported hereafter are the following:

- Thickness evaluation: f automatically set to twice the highest evaluated thickness from the (3,4)-distance transform image.
- Bounds for circular parameters: $(\theta_{min}, \rho_{min}, \rho_{max}) = (0.2rd, 15, max(w, h))$, where w, h are the image's dimensions.
- Other parameters: $m = 1$, $\sigma = 0.9999$ (4), filtering turned on.

With the exception of the images from the first contest, these default parameters are *systematically applied to all images* without any change. Our method has been implemented as a 64-bit application running on PowerPC architecture, and the results and times reported hereafter have been obtained on an Apple PowerMac dual-G5, 2.7GHz computer with 2.5GB memory. All the results, as well as an evaluation version of our method, are available for download at http://qgar.loria.fr/ranvec.

6.1 General Performances

To give an overall idea of the performances our method can achieve, we first considered the set of images from the third contest. This set consists of 10 real, large images (typical resolution: $4,800 \times 4,600$ pixels) containing architectural maps and technical drawings; illustrations are available in Fig. 9. Since a minimal tuning of parameters was allowed during this contest, we slightly modified the setup described previously in the following way: For each class of images (architectural maps, technical drawings), the user

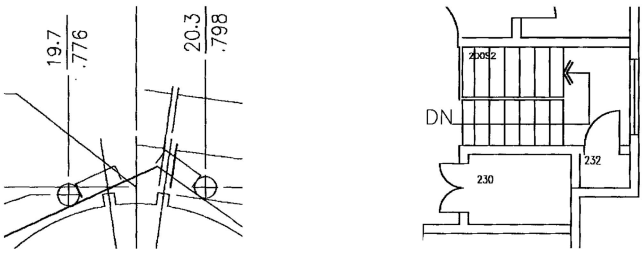


Fig. 9. Two parts of images used during the third contest.

is asked for a crude estimation of ρ_{min} and ρ_{max}; other parameters were left unchanged.

Results are presented in Table 2, obtained with the official data of the contest. This table adheres to Phillips and Chhabra's evaluation methodology [33] and has been obtained thanks to their evaluation software set with default parameters and text evaluation turned off. The processing time varies from 2mn,14s to 4mn,42s per image.

6.2 Arc Detection Capabilities

We evaluate our method's ability to extract circular arcs by following the rules of the fourth and fifth contests. The fourth contest contained three real, scanned images (p1.tif to p3.tif) and four synthetic images with Gaussian and high-frequency noise added on one side (g05.tif, f03.tif); hard-pencil, and wrapping effects on the other side (h03.tif, w03.tif). Samples are given in Fig. 11b.

Complete results are available on the left of Table 3, in which we also include results from [39] for the sake of comparison. The reader, however, may observe two important points. First, the VRI scores presented in Table 3 are different from the official results of the contest; our explanation is the following: During the contest, the circular bounds $(\theta_{min}, \rho_{min}, \rho_{max})$ for *all images* were estimated from a test image given prior to the contest, and the content of this image was also very close to that of the four synthetic images. This not only explains the very good scores obtained by all contestants for the synthetic images, but also why the average performance drops when it comes to processing real images, as retuning parameters was not allowed. Second, the

TABLE 2
Edit Cost Index (ECI) Values Obtained by the Participating Systems at the Third IAPR Contest on Graphics Recognition and by Method

Image	Mica.RW	Mica.WD	Scan2CAD	Tractrix	Vectory	VrLiu	Our method
2032_02.tif	0.699	0.644	**0.449**	0.504	0.586	0.637	0.480
2212_a.tif	0.963	0.955	0.896	0.863	0.865	0.852	**0.817**
2212_c.tif	0.897	0.891	0.616	0.682	**0.483**	0.738	0.626
2212_e.tif	0.870	0.859	0.620	0.703	**0.612**	0.713	0.693
2226_02.tif	0.703	0.681	0.474	0.621	0.465	0.561	**0.426**
pal151.tif	0.951	0.947	0.771	0.779	0.870	0.881	**0.726**
pal169.tif	0.960	0.958	0.805	0.758	0.763	0.881	**0.729**
pal311.tif	0.981	0.978	0.925	0.912	0.933	0.943	**0.902**
pal343.tif	0.956	0.953	0.829	0.809	0.871	0.886	**0.749**
pal349.tif	0.944	0.944	0.803	0.811	0.766	0.894	**0.686**
Averages	0.892	0.881	0.719	0.744	0.721	0.799	**0.683**

All these values have been computed with text evaluation turned off. The best result for each image is emphasized in bold.

TABLE 3
Left: VRI Scores Obtained for Different Systems on the Images Used During the Fourth IAPR Contest on Graphics Recognition (From Various Sources); Right: Same Results for the Fifth Contest

	syn_frq_03.tif	syn_gauss_05.tif	syn_geo_03.tif	syn_pen_03.tif	p1.tif	p2.tif	p3.tif
Elliman [14]	0.853	0.904	0.896	0.927	0.547	0.482	0.371
	Average: 0.895				Average: 0.467		
	Final score: 0.681						
Hilaire [18]	**0.889**	0.891	0.944	**0.958**	0.707	0.311	0
	Average: 0.921				Average: 0.339		
	Final score: 0.630						
Liu & Dori [52]	0.850	0.892	0.808	0.918	0.836	0.692	0.420
	Average: 0.867				Average: 0.649		
	Final score: 0.758						
Song et al. [39]	0.870	**0.951**	**0.954**	0.945	0.817	**0.747**	0.822
	Average: 0.930				Average: 0.795		
	Final score: **0.863**						
Our method	0.879	0.935	0.935	0.914	**0.845**	0.644	0.708
	Average: 0.916				Average: 0.732		
	Final score: 0.824						

Image	Elliman	Song	Our method
1.tif	0.567	0.641	**0.756**
1_230.tif	0.589	0.640	**0.752**
1_n4.tif	0.664	0.509	**0.762**
2.tif	0.439	**0.753**	0.653
2_100.tif	0.513	**0.786**	0.707
2_n4.tif	0.612	0.703	**0.791**
3.tif	0.272	0.532	**0.724**
3_100.tif	0.519	0.301	**0.740**
3_n4.tif	0.451	0.224	**0.697**
4.tif	0.500	**0.735**	0.663
4_230.tif	0.323	0.790	**0.795**
4_n4.tif	0.399	0.688	**0.782**
Average	0.487	0.609	**0.735**

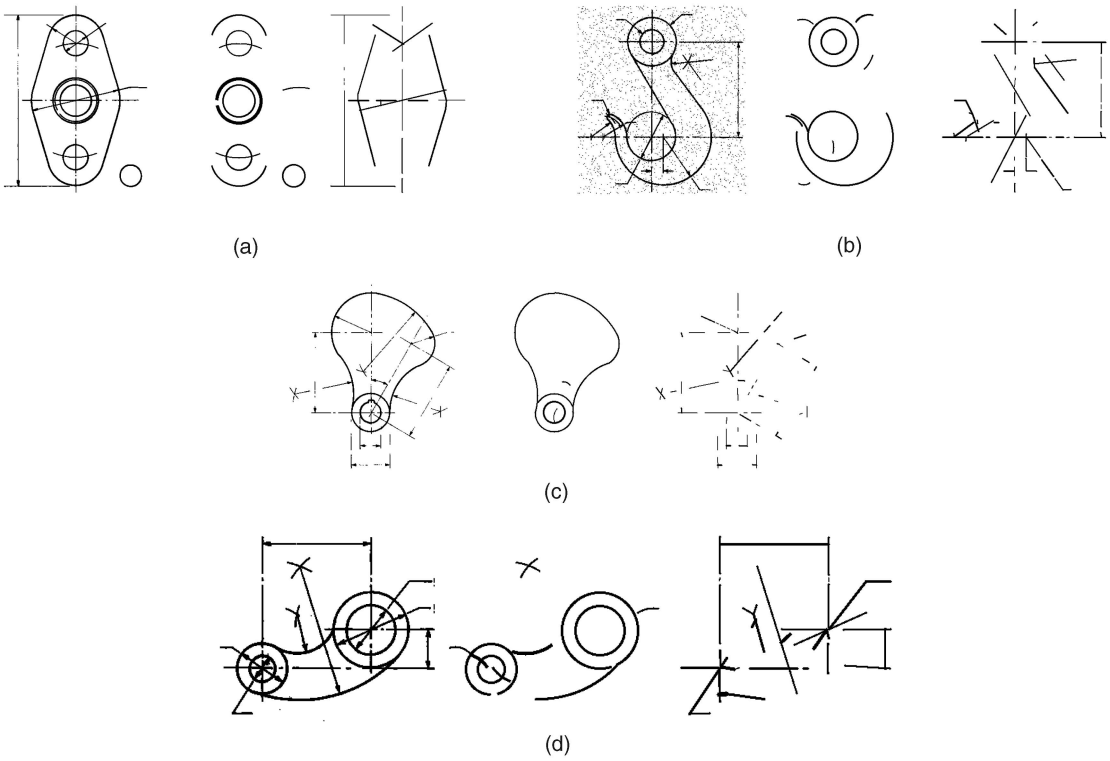


Fig. 10. Vectorization of some of the images from the fifth contest. From left to right, in each group: source image, vectorized arcs, vectorized lines; (a) image 1.tif, (b) image 2_n4.tif, (c) image 3_100.tif, and (d) image 4_230.tif.

conditions in which results have been obtained in [39] are not stated by the authors. Regarding execution times, whereas synthetic images syn_*.tif were all of very similar complexity and could be processed at around 8s each, real images p1.tif, p2.tif, and p3.tif were of unequal complexities and leaded to respective times 7s, 12s, and 1mn23s.

We applied again the same test procedure to the data of the fifth contest. The results are available at the right of Table 3 and in Fig. 10. Finally, Fig. 11 illustrates the behavior of our method with regard to the four types of noise described above. These results corroborate Lemmas 2 and 3 for Gaussian and high-frequency noise: There is a relatively good resistance for these kinds of noise, whereas the scores collapse for the two other types, either due to the disconnection of components (and, therefore, fragmentation) for hard pencil or to the m parameter set too tight to tolerate large displacements for the wrapping effects.

6.3 Robustness

A critical point for any vectorization system is its ability to distinguish text and graphical parts, as they are usually processed independently. The text/graphics separation method mentioned in Section 3.2 is able to process characters attached to graphical shapes, provided that for every text block, there exists a string seed of at least two characters not attached to any graphical shape.

However, our method is still able to eliminate text when the previous condition does not hold, provided that the length acceptance threshold f used in Section 4.2 is set at a sufficiently high value. In short, we rely on the fact that a character, once skeletonized, can't be explained by Algorithm 1 by any combination of graphical shapes with lengths all greater than f. Fig. 12a gives a first example of such a situation: A sufficiently large value of f not only permits to eliminate the text, but also removes the dummy skeleton segment remaining at the center of the X junction. A similar behavior may be observed on a complete image in Fig. 12b.

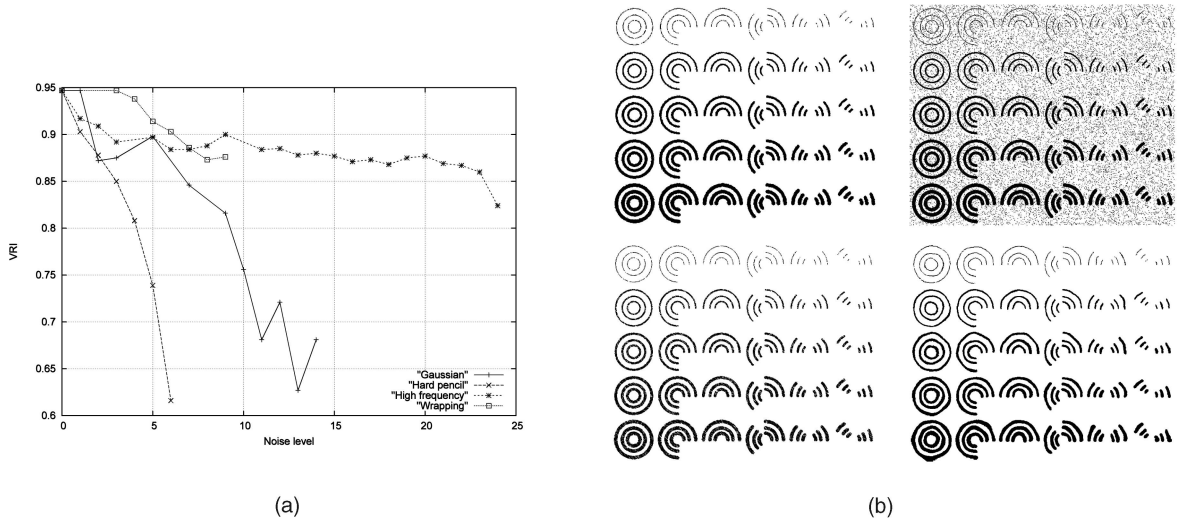
It is also noticeable that, because of the criteria used during the simplification stage (Definition 6 for simplification, (1) for unification), it is impossible that the resulting vectorization lies outside the image. This prevents precision and robustness issues; in particular, it ensures that a character remained attached to graphical part, once vectorized, creates aberrant results: At worst, its vectorization is not deleted and remains as such in the output.

6.4 Other Noticeable Details

Fig. 13 provides further details about the obtained results. Figs. 13a and 13b are two examples of correctly vectorized extracts of images from the third contest. Fig. 13c reveals an interesting point: Despite appearances, the lines in the staircase do not meet at a single point; indeed, either the staircase has been designed to fulfill this, or the drawing is simply not correct. At last, Fig. 13d shows an almost correct vectorization of the drawing: Essential parts of the shapes have been properly extracted, but small circular arcs are sometimes recognized as straight lines.

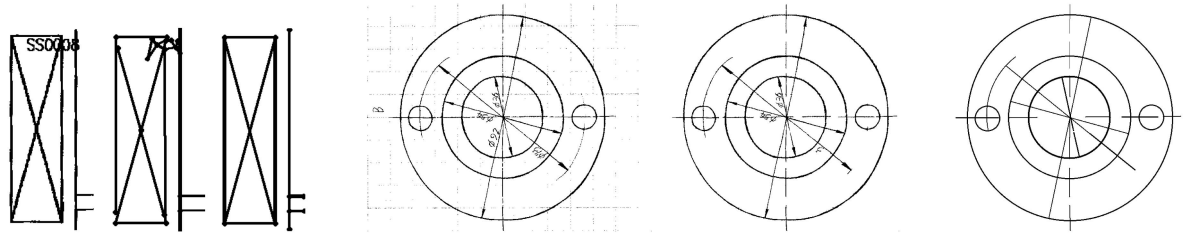
7 Conclusion

In this paper, we have presented a robust and accurate vectorization method which does not include domain knowledge and can thus be used on a variety of graphics-rich documents. The method is based on the random sampling paradigm and has the main following features:



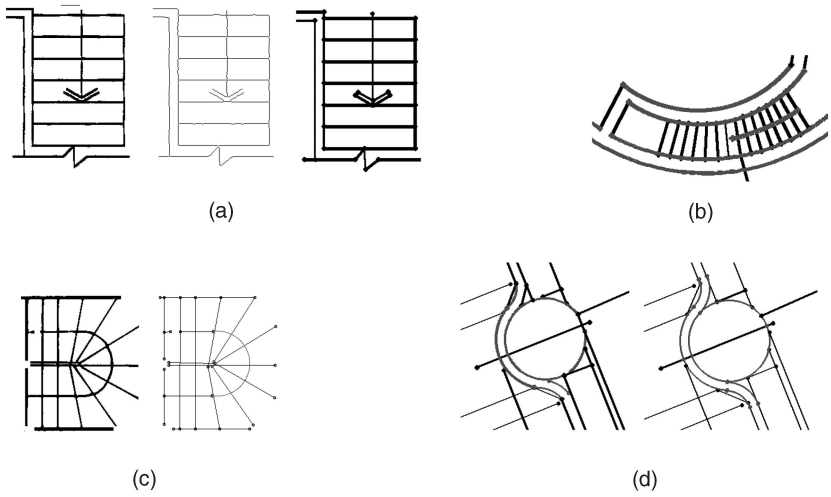
(a) (b)

Fig. 11. (a) VRI scores [29] obtained with our method for various types of noise (abscissas: noise levels as defined in [51]; ordinates: VRI scores). (b) Sample test images with no noise, Gaussian noise, hard pencil, and wrapping effects added.



(a) (b)

Fig. 12. Examples of text elimination. (a) From left to right; source image, direct vectorization (no text/graphics separation) with acceptance threshold $f = 4$, with $f = 15$. (b) From left to right: source image, result of text/graphics separation, result vectorization with $f = 20$.



(a) (b) (c) (d)

Fig. 13. A few noticeable elements properly vectorized with our vectorization method.

- it segments the skeleton into the set of the most probable graphical primitives,
- it avoids fragmentation of the curve,
- it removes fake primitives without using ad hoc rules, and
- it yields excellent precision in the positioning of the junction points.

The method is presently limited to finding straight segments and circular arcs, although it could be extended to other primitives.

Obviously, the method has higher computational complexity than simple ad hoc rules applied to the skeleton, but it remains usable; in our opinion, this is the price of the genericity and accuracy of the method.

Of course, the method still relies on a reasonably successful text-graphics segmentation and can be taken astray by characters or other noise touching the graphical parts; however, we have seen that, under certain conditions, the method remains robust in the presence of some isolated text components. Being a noncontextual method, it also fails on some specific situations (such as X junctions with very small opening angles) for which simple contextual knowledge rules would probably lead to reliable and correct graphical interpretations.

Actually, we don't think that any form of contextual knowledge should be avoided, but we feel that it should be used in complement to accurate and robust generic methods. Therefore, the proposed method can very well be used as the basis for contextual vectorization methods, in which knowledge about the graphical configurations searched for might be added to the model.

More generally, we also believe that there is room for improvement of the method itself by taking advantage of some kind of coarse-to-fine approach. The idea would be to use the present method to detect the most obvious primitives. These primitives would then be merged in a hierarchical way, starting with those having the highest confidence. The merging process can take advantage of locally available information and of contextual knowledge to perform some kind of "intelligent" reconstruction of the vector data.

ACKNOWLEDGMENTS

The authors are grateful to the reviewers for their valuable comments and suggestions on a first version of this manuscript. Special thanks also to Atul Chhabra, Ishin Phillips, and Liu Wenyin, organizers of the third, fourth, and fifth international contests on graphics recognition, for providing ground-truthed data sets and performance evaluation tools. This work was to a large extent financed by a research collaboration with FS2i, 8 impasse de Toulouse, BP 141, 78001 Versailles Cedex, France, under a CIFRE scholarship.

REFERENCES

[1] D. Antoine, S. Collin, and K. Tombre, "Analysis of Technical Documents: The REDRAW System," *Structured Document Image Analysis,* pp. 385-402, Springer Verlag, 1992.

[2] H. Asada and M. Brady, "The Curvature Primal Sketch," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 8, no. 1, pp. 2-14, 1986.

[3] H. Blum and R Nagel, "Shape Description Using Weighted Symmetric Axis Features," *Pattern Recognition,* vol. 10, pp. 167-180, 1978.

[4] E. Bodansky and M. Pilouk, "Using Local Deviations of Vectorization to Enhance the Performance of Raster-to-Vector Conversion Systems," *Int'l J. Document Analysis and Recognition,* vol. 3, no. 2, pp. 67-72, Dec. 2000.

[5] Y. Chen, N.A. Langrana, and A.K. Das, "Perfecting Vectorized Mechanical Drawings," *Computer Vision and Image Understanding,* vol. 63, no. 2, pp. 273-286, Mar. 1996.

[6] A.K. Chhabra and I.T. Phillips, "The Second International Graphics Recognition Contest—Raster to Vector Conversion: A Report," *Graphics Recognition—Algorithms and Systems,* LNCS, vol. 1389, pp. 390-410, 1998.

[7] A.K. Chhabra and I.T. Phillips, "Performance Evaluation of Line Drawing Recognition Systems," *Proc. 15th Int'l Conf. Pattern Recognition,* vol. 4, pp. 4864-4869, 2000.

[8] V. Poulain d'Andecy, J. Camillerapp, and I. Leplumey, "Kalman Filtering for Segment Detection: Application to Music Scores Analysis," *Proc. 12th Int'l Conf. Pattern Recognition,* vol. 1, pp. 301-305, 1994.

[9] G.S. di Baja, "Well-Shaped, Stable, and Reversible Skeletons from the (3,4)-Distance Transform," *J. Visual Comm. and Image Representation,* vol. 5, no. 1, pp. 107-115, 1994.

[10] D. Dori, "Orthogonal Zig-Zag: An Algorithm for Vectorizing Engineering Drawings Compared with Hough Transform," *Advances in Eng. Software,* vol. 28, no. 1, pp. 11-24, 1997.

[11] D. Dori and W. Liu, "Sparse Pixel Vectorization: An Algorithm and Its Performance Evaluation," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 21, no. 3, pp. 202-215, Mar. 1999.

[12] L. Dorst and A.W.M. Smeulders, "Discrete Straight Line Segments: Parameters, Primitives and Properties," *Vision Geometry, Series Contemporary Math.,* vol. 119, pp. 45-62, 1991.

[13] P. Dosch, P. Tombre, C. Ah-Soon, and G. Masini, "A Complete System for Analysis of Architectural Drawings," *Int'l J. Document Analysis and Recognition,* vol. 3, no. 2, pp. 102-116, Dec. 2000.

[14] D. Elliman, "TIF2VEC, An Algorithm for Arc Segmentation in Engineering Drawings," *Graphics Recognition—Algorithms and Applications,* 2002.

[15] M.A. Fischler and R.C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Comm. ACM,* vol. 24, no. 6, pp. 381-395, 1981.

[16] L.A. Fletcher and R. Kasturi, "A Robust Algorithm for Text String Separation from Mixed Text/Graphics Images," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 10, no. 6, pp. 910-918, 1988.

[17] X. Hilaire, "Segmentation Robuste de Courbes Discrètes 2D et Applications à la Rétroconversion de Documents Techniques," Thèse de Doctorat, Institut Nat'l Polytechnique de Lorraine, 2004.

[18] X. Hilaire and K. Tombre, "Improving the Accuracy of Skeleton-Based Vectorization," *Graphics Recognition—Algorithms and Applications,* 2002.

[19] O. Hori and S. Tanigawa, "Raster-to-Vector Conversion by Line Fitting Based on Contours and Skeletons," *Proc. Second Int'l Conf. Document Analysis and Recognition,* pp. 353-358, 1993.

[20] R.D.T. Janssen and A.M. Vossepoel, "Adaptive Vectorization of Line Drawing Images," *Computer Vision and Image Understanding,* vol. 65, no. 1, pp. 38-56, Jan. 1997.

[21] S.H. Joseph, "Unbiased Least Squares Fitting of Circular Arcs," *CVGIP: Graphical Models and Image Processing,* vol. 56, no. 5, pp. 424-432, Sept. 1994.

[22] S.H. Joseph and T.P. Pridmore, "Knowledge-Directed Interpretation of Mechanical Engineering Drawings," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 14, no. 9, pp. 928-940, Sept. 1992.

[23] T. Kanungo, R.M. Haralick, H.S. Baird, W. Stuezle, and D. Madigan, "A Statistical, Nonparametric Methodology for Document Degradation Model Validation," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 22, no. 11, pp. 1209-1223, Nov. 2000.

[24] R. Kasturi, S.T. Bow, W. El-Masri, J. Shah, J.R. Gattiker, and U.B. Mokate, "A System for Interpretation of Line Drawings," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 12, no. 10, pp. 978-992, Oct. 1990.

[25] P. Kultanen, E. Oja, and L. Xu, "Randomized Hough Transform (RHT) in Engineering Drawing Vectorization System," *Proc. IAPR Workshop Machine Vision Applications,* pp. 173-176, 1990.

[26] L. Lam, S.-W. Lee, and C.Y. Suen, "Thinning Methodologies—A Comprehensive Survey," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 14, no. 9, pp. 869-885, Sept. 1992.

[27] K.H. Lee, Y.C. Choy, and S.B. Cho, "Geometric Structure Analysis of Document Images: A Knowledge-Based Approach," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 22, no. 12, pp. 1224-1240, Dec. 2000.

[28] W. Liu, "Report of the Arc Segmentation Contest," *Graphics Recognition: Recent Advances and Perspectives,* 2004.

[29] W.Y. Liu and D. Dori, "A Protocol for Performance Evaluation of Line Detection Algorithms," *Machine Vision and Applications,* vol. 9, nos. 5-6, pp. 240-250, 1997.

[30] F. Mokhatarian and A.K. Mackworth, "A Theory of Multiscale, Curvature-Based Shape Representation for Planar Curves," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 14, no. 8, pp. 789-805, Aug. 1992.

[31] V. Nagasamy and N.A. Langrana, "Engineering Drawing Processing and Vectorization System," *Computer Vision, Graphics and Image Processing,* vol. 49, no. 3, pp. 379-397, 1990.

[32] J. O'Rourke, S.R. Kosaraju, and N. Meggido, "Computing Circular Separability," *Discrete and Computational Geometry,* vol. 1, pp. 105-113, 1986.

[33] I.T. Phillips and A.K. Chhabra, "Empirical Performance Evaluation of Graphics Recognition Systems," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 21, no. 9, pp. 849-870, Sept. 1999.
[34] U. Ramer, "An Iterative Procedure for the Polygonal Approximation of Plane Curves," *Computer Graphics and Image Processing,* vol. 1, pp. 244-256, 1972.
[35] P.L. Rosin, "Techniques for Assessing Polygonal Approximation of Curves," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 19, no. 6, pp. 659-666, June 1997.
[36] P.L. Rosin and G.A. West, "Segmentation of Edges into Lines and Arcs," *Image and Vision Computing,* vol. 7, no. 2, pp. 109-114, May 1989.
[37] R.W. Smith, "Computer Processing of Line Images: A Survey," *Pattern Recognition,* vol. 20, no. 1, pp. 7-15, 1987.
[38] J. Song and M.R. Lyu, "A Hough Transform Based Line Recognition Method Utilizing Both Parameter Space and Image Space," *Pattern Recognition,* vol. 38, no. 4, pp. 539-552, Apr. 2005.
[39] J. Song, M.R. Lyu, and S. Cai, "Effective Multiresolution Arc Segmentation: Algorithms and Performance Evaluation," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 26, no. 11, pp. 1491-1506, Nov. 2004.
[40] J. Song, F. Su, C.-L. Tai, and S. Cai, "An Object-Oriented Progressive-Simplification Based Vectorization System for Engineering Drawings: Model, Algorithm, and Performance," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 24, no. 8, pp. 1048-1060, Aug. 2002.
[41] C.-H. Teh and R.T. Chin, "On the Detection of Dominant Points on Digital Curves," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 11, no. 8, pp. 859-872, Aug. 1989.
[42] E. Thiel, "Les Distances de Chanfrein en Analyse d'Image: Fondements et Applications," Thèse de Doctorat, Université Joseph Fourier—Grenoble I, 1994.
[43] K. Tombre, C. Ah-Soon, P. Dosch, G. Masini, and S. Tabbone, "Stable and Robust Vectorization: How to Make the Right Choices," *Graphics Recognition—Recent Advances,* Sept. 2000.
[44] "Graphics Recognition—Algorithms and Systems," *Lecture Notes in Computer Science,* K. Tombre and A.K. Chhabra, eds., Apr. 1998.
[45] K. Tombre, S. Tabbone, L. Pélissier, B. Lamiroy, and P. Dosch, "Text/Graphics Separation Revisited," *Proc. Fifth IAPR Int'l Workshop Document Analysis Systems,* Aug. 2002.
[46] P. Veelaert, "Concurrency of Line Segments in Uncertain Geometry," *Proc. 10th Int'l Conf. Discrete Geometry for Computer Imagery,* 2002.
[47] K. Wall and P. Danielsson, "A Fast Sequential Method for Polygonal Approximation of Digitized Curves," *Computer Vision, Graphics and Image Processing,* vol. 28, pp. 220-227, 1984.
[48] L. Wenyin and D. Dori, "A Survey of Non-Thinning Based Vectorization Methods," *Advances in Pattern Recognition (Proc. Joint IAPR Workshops SSPR '98 and SPR '98,* Aug. 1998.
[49] L. Wenyin and D. Dori, "Incremental Arc Segmentation Algorithm and Its Evaluation," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 20, no. 4, pp. 424-431, Apr. 1998.
[50] L. Wenyin, J. Zhai, and D. Dori, "Extended Summary of the Arc Segmentation Contest," *Graphics Recognition—Algorithms and Applications,* LNCS, vol. 2390, pp. 273-288, 2002.
[51] L. Wenyin, J. Zhai, D. Dori, and T. Long, "A System for Performance Evaluation of Arc Segmentation Algorithms," *Proc. CVPR Workshop Empirical Evaluation in Computer Vision,* Dec. 2001.
[52] L. Wenyin and D. Dori, "Genericity in Graphics Recognition Algorithms," *Lecture Notes in Computer Science,* K. Tombre and A.K. Chhabra, eds., pp. 9-20, Apr. 1998.
[53] H. Yamada, *Paper-Based Map Processing, Handbook of Character Recognition and Document Image Processing,* chapter 19, pp. 503-528. World Scientific, 1997.
[54] Y. Zheng, H. Li, and D. Doermann, "A Parallel-Line Detection Algorithm Based on HMM Decoding," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 27, no. 5, pp. 777-792, May 2005.

Xavier Hilaire received the PhD degree from the Institut National Polytechnique de Lorraine in early 2004 on the topic of robust curve segmentation and document analysis. He was then a postdoctoral fellow from the Lavoisier programme (French Ministry of Foreign Affairs) at the University of Auckland, New Zealand. In 2005, he was a temporary assistant professor at Université Henri Poincaré, Nancy, France. He is currently a reseach fellow at the Department of Computing Science, University of Glagow, United Kingdom. His research interests include discrete geometry, document analysis, pattern recognition, and computer vision.

Karl Tombre is a professor at Ecole des Mines de Nancy (Institut National Polytechnique de Lorraine) since 1998. From 1987 to 1998, he was senior researcher at INRIA. His main research topic is in the realm of document analysis and graphics recognition. Professor Tombre is the scientific leader of the QGAR research project at the LORIA research center and at INRIA Lorraine. He is currently president of the French Association for Pattern Recognition and Image Analysis (AFRIF) and first vice president of the International Association for Pattern Recognition (IAPR). He is editor-in-chief of the *International Journal on Document Analysis and Recognition*, a member of the advisory board of ELCVIA, MGV, and ARIMA, and member of numerous conference and workshop committees.